NEOBOTIX

thinking forward

USBoard-USS5

Neobotix GmbH

Nov 11, 2026

User Manual

1 Introduction **1**
- 1.1 About This Documentation . . . 1
  - 1.1.1 Symbols and Conventions . . . 1
  - 1.1.2 Further Information . . . 2
- 1.2 Legal Notes . . . 2
  - 1.2.1 Version Information . . . 2
  - 1.2.2 Liability . . . 2
  - 1.2.3 Downloads and Further Information . . . 2

2 USBoard-USS5 **3**
- 2.1 Introduction . . . 4
  - 2.1.1 Intended Use . . . 4
  - 2.1.2 Improper Use . . . 4
  - 2.1.3 Qualified Personnel . . . 4
- 2.2 Operating Principle . . . 4
  - 2.2.1 Basic Principle . . . 4
  - 2.2.2 Cross Echo Mode . . . 5
- 2.3 Technical Data . . . 6
  - 2.3.1 USBoard-USS5 . . . 6
  - 2.3.2 Ultrasonic Sensor Bosch USS5 . . . 6
  - 2.3.3 USS5 Field of View . . . 7
- 2.4 Parameter Set . . . 9
- 2.5 Command Set . . . 11
  - 2.5.1 Command IDs . . . 11
  - 2.5.2 CAN Communication . . . 11
  - 2.5.3 USB / Serial Interface . . . 12
- 2.6 Commands . . . 13
  - 2.6.1 CMD_CONNECT . . . 13
  - 2.6.2 CMD_SET_CHANNEL_ACTIVE . . . 13
  - 2.6.3 CMD_GET_DATA_1TO8 . . . 14
  - 2.6.4 CMD_GET_DATA_9TO16 . . . 14
  - 2.6.5 CMD_WRITE_PARASET . . . 14
  - 2.6.6 CMD_WRITE_PARASET_TO_EEPROM . . . 15
  - 2.6.7 CMD_READ_PARASET . . . 15
  - 2.6.8 CMD_GET_ANALOGIN . . . 16
  - 2.6.9 CMD_GET_DATA . . . 16

2.7 ROS Node . . . 17
2.7.1 Installation . . . 17
2.7.2 Launch . . . 18
2.7.3 Parameters . . . 18
2.7.4 Topics . . . 19
2.7.5 Multiple USBoards . . . 19
2.7.6 Help . . . 20
2.8 ROS 2 Node . . . 20
2.8.1 Installation . . . 20
2.8.2 Launch . . . 21
2.8.3 Parameters . . . 21
2.8.4 Topics . . . 22
2.8.5 Help . . . 22
2.9 Graphical User Interface . . . 22
2.9.1 Introduction . . . 22
2.9.2 First Steps . . . 23
2.9.3 Configuration . . . 25
2.10 Dimensions . . . 29
2.11 Connector Descriptions . . . 30
2.11.1 X1 . . . 30
2.11.2 X2 - X5 . . . 30
2.11.3 X6 . . . 31
2.11.4 X7 . . . 31
2.11.5 USS5 . . . 31
2.12 Additional Parts . . . 31
2.12.1 Connectors . . . 31
2.12.2 USBoard-USS5 Configuration Cable . . . 31
2.12.3 Sensor Cable Sets . . . 32
2.12.4 Connector Sets . . . 32
2.12.5 Ultrasonic Sensors . . . 32
2.12.6 USS5 Mounting Frames . . . 35
2.13 FAQ . . . 35
2.13.1 General Questions . . . 36
2.13.2 Connection Problems . . . 36
2.14 Legal Notes . . . 36
2.14.1 EU Declaration of Conformity . . . 37
2.14.2 RoHS Information . . . 37

3 Connectors **38**
3.1 TE Connectivity - HE14 . . . 38
3.2 Würth Elektronik - MPC4 . . . 39
3.3 Würth Elektronik - MPC3 . . . 40

4 Qualified Personnel **42**

CHAPTER 1

Introduction

1.1 About This Documentation

1.1.1 Symbols and Conventions

The following symbols and highlighting are used in this documentation:

Danger: Indicates a hazardous area or an immediately dangerous situation that could lead to serious injury or even death.

Warning: Indicates a hazardous area or a potentially dangerous situation that could lead to serious injury or damages.

Attention: Indicates hazards or situations that can lead to minor injuries, damages or other negative effects.

Note: Indicates important information that must be observed for safe operation.

Tip: Provides helpful tips to make working with the mobile robot easier and more efficient.

- Lists contain several items of information on the same topic.
- Where possible and appropriate, they are prioritised, with the most important entry at the top.
- Lists do not claim to be exhaustive unless otherwise stated.

1. Instructions are numbered.

2. Numbered instructions must be followed in the given order.

1.1.2 Further Information

Further information, especially on customised configurations and systems, will be provided with your robot on delivery or on request[1]. In most cases, all documents relating to your customised robot are also available in the download area[2] of our website.

1.2 Legal Notes

1.2.1 Version Information

The English part of this online documentation has been translated and is not the original. Please refer to the German version in case of uncertainties or questions.

1.2.2 Liability

Every care has been taken in the preparation of this manual which represents the state of technology at the time of its composing. However, inaccuracies or omissions might occur. Please inform Neobotix in case you notice any.

The Neobotix GmbH cannot be held responsible for any technical or typographical errors and reserves the right to make changes to the product and manual without prior notice. Neobotix makes no warranty of any kind with regard to the material contained within this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Neobotix GmbH shall not be liable or responsible for incidental or consequential damages in connection with the improper use of one or more of the products described in this manual.

1.2.3 Downloads and Further Information

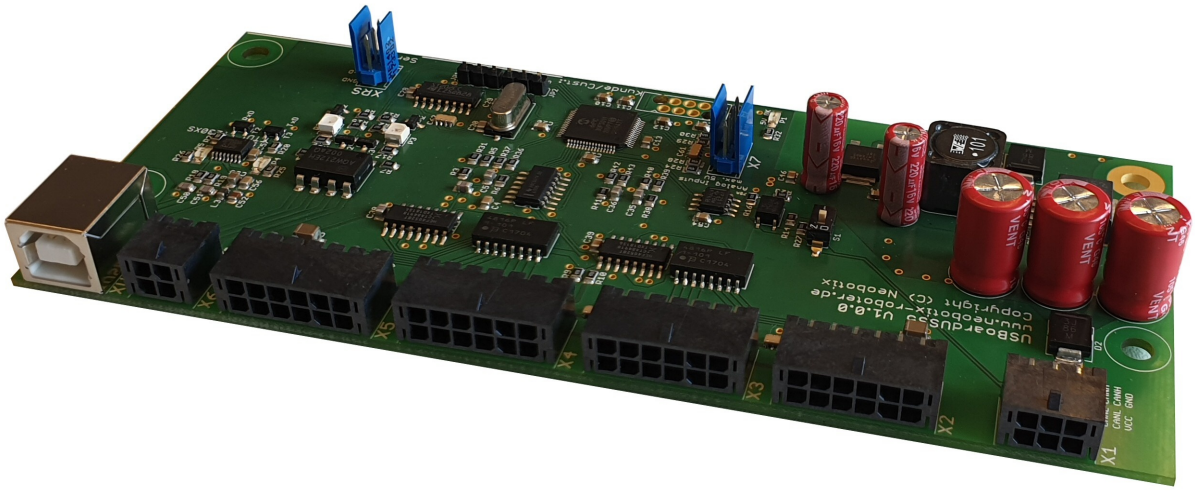
Additional information, data sheets and documentations, also for other products of Neobotix, can be found in the download section on our homepage: https://www.neobotix-robots.com/service/downloads.

[1] https://www.neobotix-robots.com/contact/contact-details
[2] https://www.neobotix-roboter.de/login

CHAPTER 2

USBoard-USS5

↓ Download as PDF[3]



Ultrasonic sensors are used to measure the distance to objects within the sensor's metering range. The metering principle is based on measuring the time of flight of a sound pulse. This pulse is created by the sensor, reflected by an obstacle and then received by the sensor.

The USBoard-USS5 is capable of operating up to 16 ultrasonic sensors of type Bosch USS5. It can be used for collision detection on autonomous vehicles like mobile robot, wheeled drones and other AGVs. The board features easy commissioning, comfortable parametrization and a wide range of custom settings. Defining warning and alarm distances allows an easy detection of possible collisions.

Furthermore four analogue inputs are available on the board. These inputs can be used to add other sensors to the system without the need for any additional boards.

[3] https://neobotix-docs.de/hardware/en/USBoard-USS5.pdf

Tip: The Graphical User Interface for the USBoard-USS5 can be found at *Graphical User Interface* (page 22).

2.1 Introduction

See also our *FAQ* (page 35).

2.1.1 Intended Use

The USBoard-USS5 was designed for use in mobile robots and similar machines like wheeled drones or AGVs. It is meant for obstacle detection and collision avoidance.

The USBoard-USS5 can also be used in stationary machines or equipment to provide distance measurement data.

The USBoard-USS5 was developed to provide exclusively non-safe data and information.

2.1.2 Improper Use

The USBoard-USS5 is explicitely **not a safety device** and must not be used as such. It must never be used to safeguard dangerous areas or as single system for collision avoidance.

It is not recommended to use the USBoard-USS5 in applications with very high requirements on measurement accuracy, on reliability of the measurements or on data output speed. This for example includes motion control applications like parking.

2.1.3 Qualified Personnel

The USBoard-USS5 must only be installed, connected and brought into service by *qualified personnel* (page 42).

2.2 Operating Principle

2.2.1 Basic Principle

An ultrasonic sensor creates a sound pulse that is reflected by obstacles and then returns to the sensor. By measuring the time between emitting and receiving the ultrasonic pulse, the USBoard-USS5 is able to calculate the distance between the sensor and the closest obstacle.

In default mode the sensors are operated cyclically one after. In *cross echo mode* (page 5) the sensors operate in groups of four. Only one of these sensors emits the pulse but all sensors can receive the echo. If the geometric positions of the sensors are know it is possible to triangulate the position of the detected obstacle.

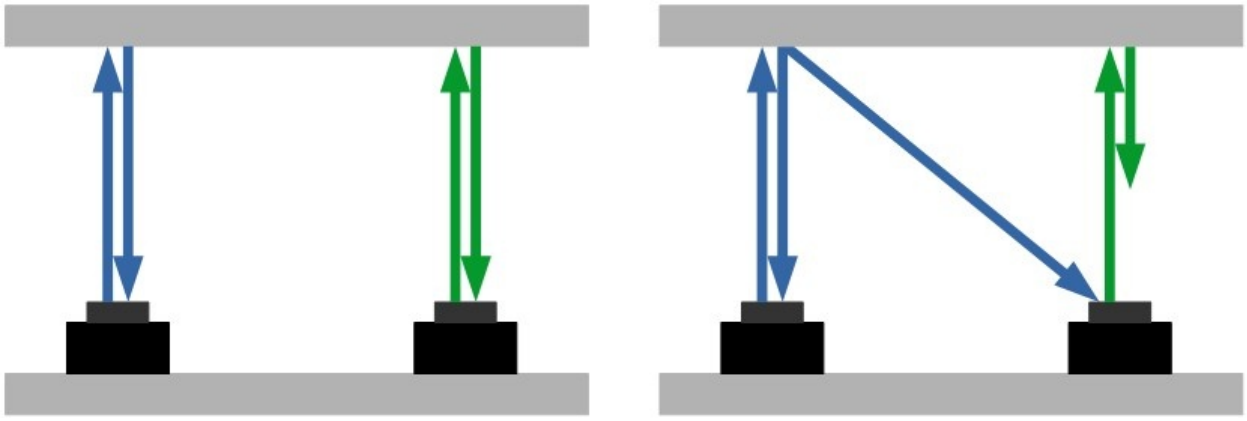
A CAN and a USB communication interface can be used to acquire distance data measured by the sensors as well as the data from the analogue inputs. On request the USB port can be replaced by a RS-232 interface.

For each sensor a warning and an alarm distance can be defined by software. This allows using the board for collision protection. As soon as the distance measured by a sensor falls below the according warning distance, an LED on the board lights up and a relay is switched. A second LED and another relay indicate that an obstacle was detected within the alarm distance of one or more sensors. Warning and alarm distances are defined in the parameter set via the GUI.

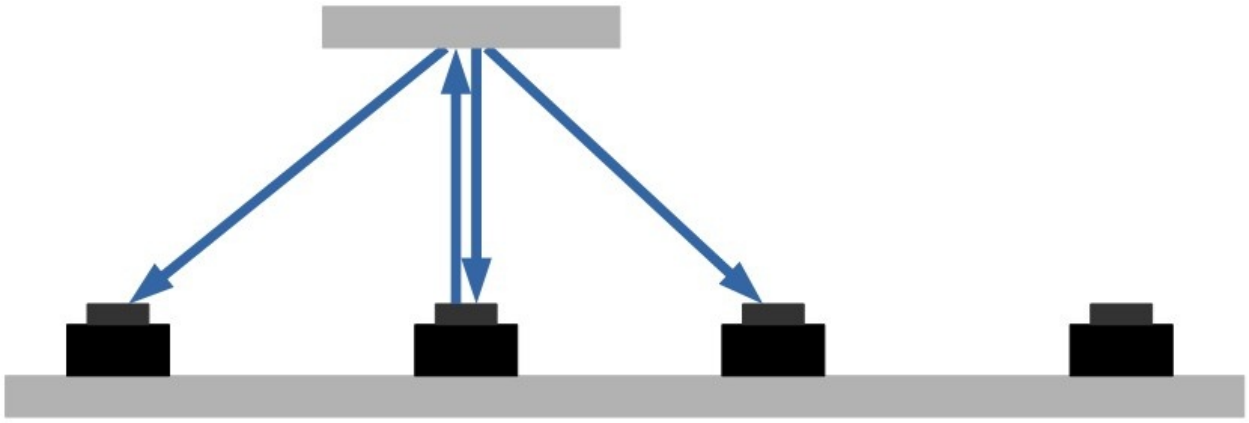
Warning: The USBoard-USS5 is no safety system and can only provide supporting, non-safe information. Never use the USBoard-USS5 for safeguarding of dangerous areas or movements.

2.2.2 Cross Echo Mode

In the traditional, sequential operation mode the sensors are independent from each other. They operate in an infinite loop one after the other, with each sensor sending its own ultrasonic pulse and waiting for the echo to calculate the distance to the object (left picture below). If a sensor receives an echo from a pulse that was sent earlier by another sensor it will be treated like a regular echo, resulting in a faulty measurement (right picture).



In cross-echo mode only one sensor of the group of four will send a pulse but all four sensors can receive the echo. If the positions of all sensors are known it is possible to estimate the location of the detected object based on the different durations it takes the echo to reach each sensor.



Several aspects should be kept in mind when calculating the object's position.

- Depending on the shape and surface of an object the ultrasound pulse is reflected or absorbed differently. This is why some objects may be detectable with direct measurements in normal mode but do not create a sufficiently strong cross-echo.
- The realistic field of view of the sensors is rather limited at large distances and also dependent on form and material of the object. It is thus recommended to do some experiments and tests in cross-echo mode before setting up the actual application.
- The distance calulation performed by the USBoard-USS5 is the same in both operating modes: The time between sending and receiving the echo is halved and multiplied with the speed of sound. This reduces processing

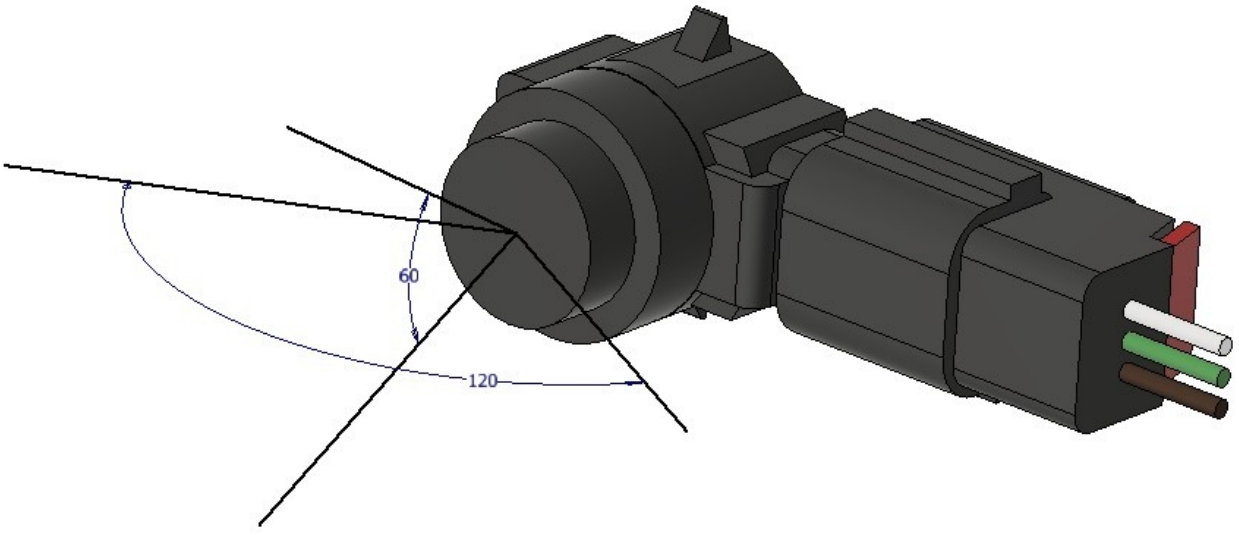
and cycle times and makes the data output easier. If the distance between the sensors of a cross-echo group is not too big this simplified calculation often provides usable results, despite being not perfectly accurate. For an exact triangulation please double the measurement output to get the total distance from the sending sensor to the obstacle and back to each sensor.

2.3 Technical Data

2.3.1 USBoard-USS5

- Supply voltage from +9 VDC up to +60 VDC, max. 4 W
- Digital communication via CAN and USB, optional RS-232
- Optical indicators for warning and alarm distance
- Solid state relay outputs for warning and alarm distance (max. 60 V, 0,5 A)
- Four analogue inputs (0 V – 5 V)
- Weight: 71 g
- Storage temperature range: -40 °C – 85 °C
- Operation temperature range: 0 °C – 70 °C
- Customs tariff number: 9031 9000
- Order number: X110

2.3.2 Ultrasonic Sensor Bosch USS5



- Theoretical field of view: 120° x 60°, symmetrical
- Realistic field of view at maximum distance: 10° x 10°, symmetrical
- Please check our FOV test results *below* (page 7).
- Operating frequency: 48 kHz
- Range: 0.2 m up to 3.3 m

- Detection of objects closer than the minimum distance
- Indication of empty measurement range (no echo)
- Customs tariff number: 9031 8080
- Order number: X210

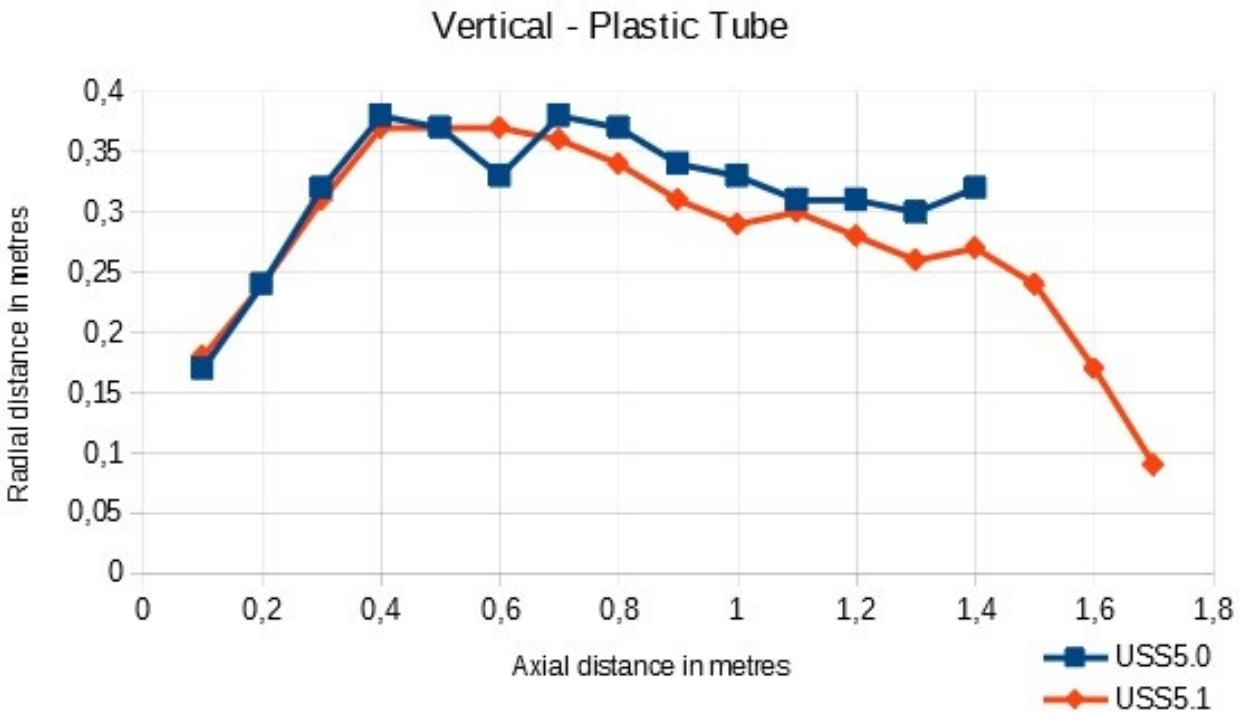
2.3.3 USS5 Field of View

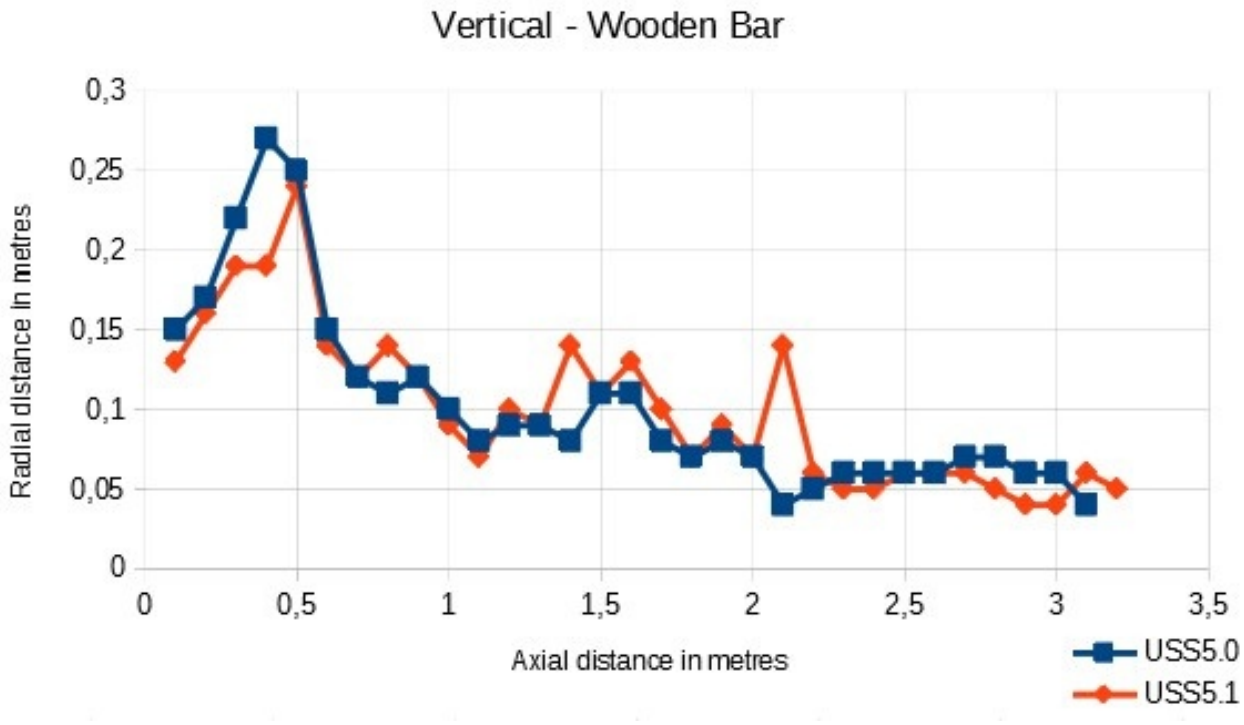
We have tested the USS5 real world field of view with different objects and configurations. So far we did tests with

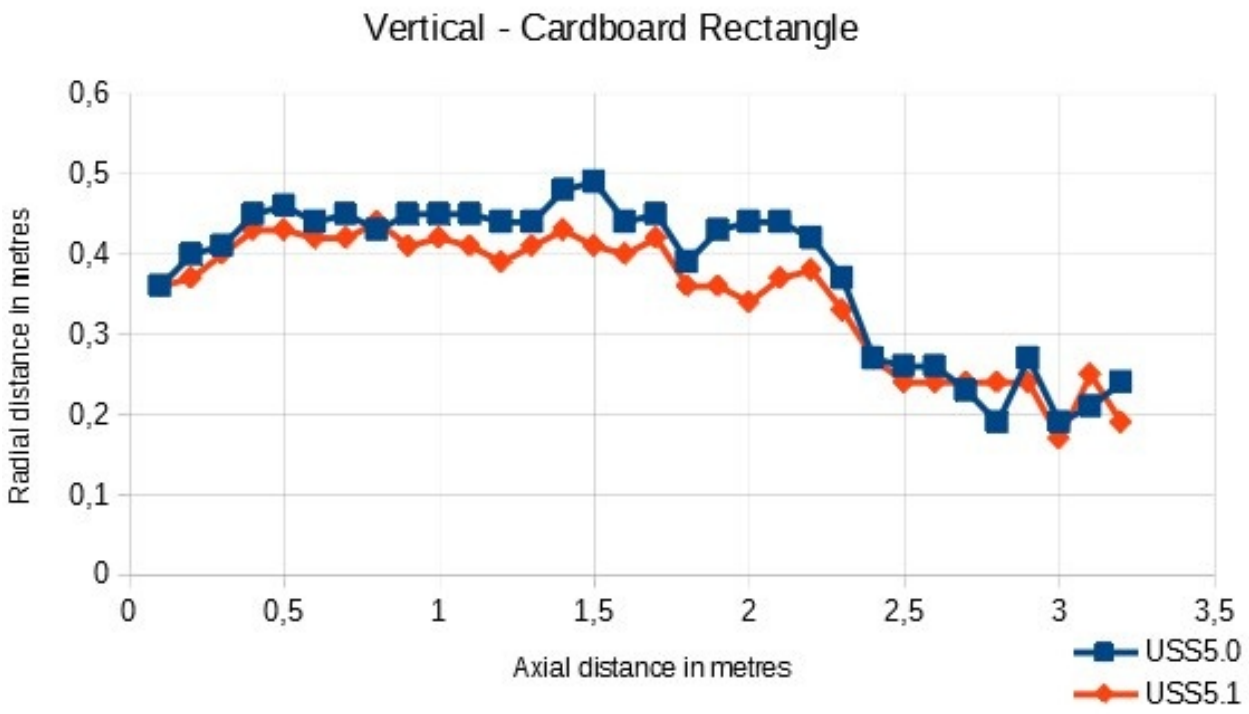
- a plastic tube (diameter 75 mm, height 1.75 m)
- a wooden bar (width 0.1 m, height 1.77 m)
- a cardboard rectangle (width 0.6 m, height 0.4 m)
- a human (height 1.8 m)

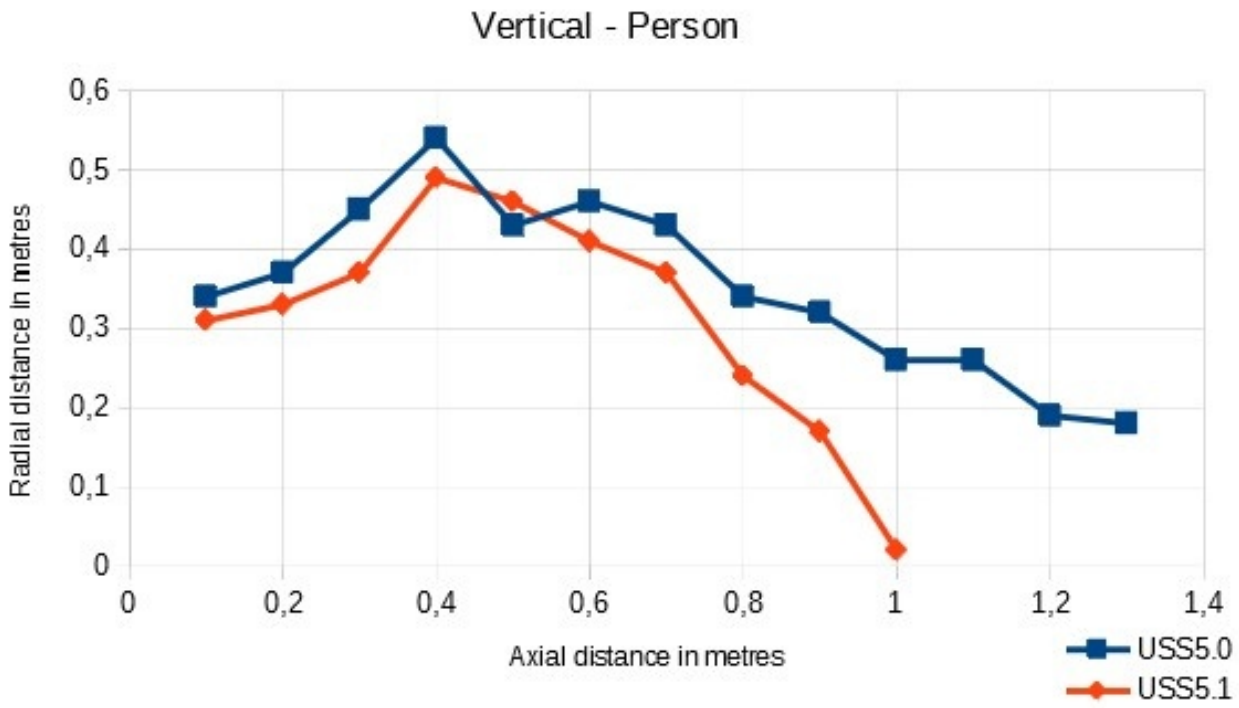
You can `download our full data set here`.

Note: Please note that the data on this page are only for reference and are no assurance of any product features. Please perform your own tests to check the suitability of the USBoard-USS5 and the USS5 sensors for your specific application.




Vertical - Wooden Bar
Radial distance in metres
0,3
0,25
0,2
0,15
0,1
0,05
0
0
0,5
1
1,5
2
2,5
3
3,5
Axial distance in metres
USS5.0
USS5.1


Vertical - Cardboard Rectangle
Radial distance in metres
0,6
0,5
0,4
0,3
0,2
0,1
0
0
0,5
1
1,5
2
2,5
3
3,5
Axial distance in metres
USS5.0
USS5.1



2.4 Parameter Set

The USBoard-USS5 uses a parameter set which allows a custom configuration. The graphical user interface can be used to easily manipulate the board's settings, store them on the host computer's hard drive, write them to the EEPROM on the USBoard-USS5 or read the current values from the board.

The parameter set consists of 54 bytes as described below.

Note: Several features affect groups of sensors. In these cases group 0 contains sensors 1 to 4, group 1 contains sensors 5 to 8 and so on.

Byte 0 CAN baud rate

0	1000 kBaud (default)
1	500 kBaud
2	250 kBaud
3	125 kBaud
4	100 kBaud
5	50 kBaud

Bytes 1-4 CAN base address (default `0x400`), calculated as follows:

```
(byte_4 << 24) | (byte_3 << 16) | (byte_2 << 8) | byte_1
```

Byte 5 Flags as bitmask

Bit	Meaning
0	Enable CAN extended id (default 0)
1	Enable CAN termination of USBoard-USS5-IP (default 0)
2	Enable analog input (default 0)
3	Enable legacy data format (default 0)
4	Warning relay blocked mode (default 0) (see *Configuration* (page 25))
5	Alarm relay blocked mode (default 0) (see *Configuration* (page 25))

In legacy mode, the data format is the same as the previous version of the USBoard, compatible to RelayBoardV2. This only affects continous transmission mode.

Byte 6 Bits 0 to 3 encode the transmission mode for measurement data.

0	Send on request
1	Send continously via CAN
2	Send continously via USB / RS-232
3	Send continously via CAN and USB / RS-232

Bits 4 to 7 select which group will send measured data (bit 4 for group 0, bit 5 for group 1 and so on). If all bits are set to 0, all groups will send.

Default is `0xF0`, i.e. all groups transmit on request.

Byte 7 Bits 0 to 3 encode the transmission interval when sending continously.

0	0.5 s
1	1.0 s
2	2.0 s
3	0.2 s
15	custom

Bits 4 to 7 encode a custom transmission interval in 50 ms increments (0 for 50 ms, 1 for 100 ms and so on).

Byte 8 Sensors 1 to 8 active, bit coded. Default is `0xFF`, i.e. all active.

Deactivated sensors do not send ultrasonic pulses. Deactivating sensor channels that are not in use reduces the overall cycle time.

Byte 9 Sensors 9 to 16 active, bit coded. Default is `0xFF`, i.e. all active.

Deactivated sensors do not send ultrasonic pulses. Deactivating sensor channels that are not in use reduces the overall cycle time.

Bytes 10-25 Warning distances of sensors 1-16, in cm. Default is 100 cm.

Bytes 26-41 Alarm distances of sensors 1-16, in cm. Default is 30 cm.

Byte 42 Sensor resolution, 2 bits per group

0	1 cm
1	0.5 cm (default)
2	0.25 cm
3	0.125 cm

Byte 43 Enable cross echo mode, one bit per group. Default is 0.

Byte 44 Cross echo configuration, 2 bits per group denoting the sending sensor (0 to 3). Default is 0, i.e. the first sensor in the group.

Bytes 45-46 Fire interval in increments of 10 ms, 4 bits per group (0 for 10 ms, 1 for 20 ms and so on). Default is 20 ms.

Byte 47 Low-pass filter gain (see below) for distance measurements, in 0.0078125 increments. Default is 128, i.e. gain 1, filtering disabled.

Byte 48 Reserved, set to arbitrary values when sending.

Byte 49 Hardware version, read only.

20	USBoard-USS5 V1.0.0
21	USBoard-USS5 V1.0.1
200	USBoard-USS5-IP V1.0.0
201	USBoard-USS5-IP V1.0.1

Bytes 50-53 Serial number, read only.

(In USBoard V1, only bytes 51-53 are used.)

The low pass filter feature can be used smoothen the measurements and to prevent single, potentially faulty, readings from immediately switching the relays. Each new measurement of a sensor is added to the output value according to the filter's weight.

A setting of 0.3 means that the output is calculated from 70% of the previous output value and 30% of the new measurement.

A setting of 1.0 means that each new measurement is directly used as output value while the old value is discarded, in effect deactivating the filter.

2.5 Command Set

2.5.1 Command IDs

The following table shows the available commands of the USBoard-USS5.

Command	Value	Description
`CMD_CONNECT`	0	Check connection
`CMD_SET_CHANNEL_ACTIVE`	1	Activate channel for sending / receiving
`CMD_GET_DATA_1TO8`	2	Request data of sensors 1 to 8
`CMD_GET_DATA_9TO16`	3	Request data of sensors 9 to 16
`CMD_WRITE_PARASET`	4	Upload parameter set to board (volatile)
`CMD_WRITE_PARASET_TO_EEPROM`	5	Write parameter set to board's EEPROM (non-volatile)
`CMD_READ_PARASET`	6	Read current parameter set
`CMD_GET_ANALOGIN`	7	Read values of analogue inputs
–	8-12	Reserved
`CMD_GET_DATA`	13	Request data of all sensors

2.5.2 CAN Communication

The default base address is `0x400`. This base address can be changed in the parameter set.

The addresses used by the USBoard-USS5 are calculated from the base address by adding the following offsets.

Offset to base address	Function
+0	Receive commands
+1	Answer to CMD_CONNECT
+2	First answer to CMD_GET_DATA_1TO8
+3	Second answer to CMD_GET_DATA_1TO8
+4	First answer to CMD_GET_DATA_9TO16
+5	2nd answer to CMD_GET_DATA_9TO16
+6	Answer to CMD_READ_PARASET
+7	Answer to CMD_GET_ANALOGIN
+8	Answer to CMD_WRITE_PARASET
+9	Answer to CMD_WRITE_PARASET_TO_EEPROM
+13	1st Answer to CMD_GET_DATA (group 0: 1 to 4)
+14	2nd Answer to CMD_GET_DATA (group 1: 5 to 8)
+15	3rd Answer to CMD_GET_DATA (group 2: 9 to 12)
+16	4th Answer to CMD_GET_DATA (group 3: 13 to 16)

2.5.3 USB / Serial Interface

Communication via USB is done via a USB-serial-converter on the USBoard-USS5. The computer's operating system can access it like a traditional serial interface. On request the USB converter can be replaced by an RS-232 transmitter. This does not affect the communication protocol of the serial interface.

Note: The RS-232 interface requires a common ground connection between the USBoard-USS5 and the computer.

The serial interface runs at 19200 Baud. Its protocol has the same format as the CAN communication except for the following addition:

Every message **from** the USBoard-USS5 begins with a start byte which must have the value 0xFF. After each message, made up of eight data bytes, a 16 bit checksum is sent. Because of this, a complete message is 11 bytes long.

Byte 1	Bytes 2-9	Byte 10	Byte 11
0xFF	data bytes	checksum high byte	checksum low byte

The checksum is calculated with an CRC-CCITT algorithm using 8 databytes.

Messages **to** the USBoard-USS5 only contain the data bytes.

Example implementation of the checksum calulation (C code):

```
unsigned int getCheckSum(unsigned char *c, size_t iNumBytes){
        unsigned int uCrc16;
        unsigned char ucData[2];
        size_t i;

        uCrc16 = 0;
        ucData[0] = 0;
        for(i=0; i<iNumBytes; i++){
                ucData[1] = ucData[0];
                ucData[0] = c[i];
```

(continues on next page)

(continued from previous page)

```
                if(uCrc16 & 0x8000){
                        uCrc16 = (uCrc16 & 0x7fff) << 1;
                        uCrc16^= 0x1021;
                }else{
                        uCrc16 <<= 1;
                }
                uCrc16^= (unsigned int)(ucData[0]) | ((unsigned int)(ucData[1]) << 8);
        }

        return uCrc16;
}
```

2.6 Commands

You can find the command set at *Command Set* (page 11).

Note: Byte 0 in this documentation refers to the first data byte after the header. Individual bytes are transferred with the Least Significant Bit first.

In the following, the CAN IDs are given as an offset to the base address, so `+3` means base address plus 3.

2.6.1 CMD_CONNECT

Use this command to establish and check the connection to the board.

Command ID: `+0`

CMD_CONNECT	0	0	0	0	0	0	0

Answer ID: `+1`

CMD_CONNECT	1	2	3	4	5	6	7

2.6.2 CMD_SET_CHANNEL_ACTIVE

This command can be used to activate or deactivate individual sensors without transmitting the complete parameter set. Two bytes for the channels 1 to 8 and 9 to 16 contain the information whether the sensors should be active. These bytes are bit-coded, each bit representing the state of one sensor. Every active channel is marked with a 1.

For example, `0x1F` as the first byte means channels 1 to 5 are active and channels 6 to 8 are not active.

Command ID: `+0`

CMD_SET_CHANNEL_ACTIVE	(Sensors 1 to 8)	(Sensors 9 to 16)	0	0	0	0	0

Answer No answer.

2.6.3 CMD_GET_DATA_1TO8

This command is used to request the readings of sensors 1 to 8.

Command ID: `+0`

CMD_GET_DATA_1TO8	0	0	0	0	0	0	0

Answer (two parts) IDs: `+2`, `+3`

CMD_GET_DATA_1TO8	0	Values for sensors 1-4 (one byte each)	0	(reserved)
CMD_GET_DATA_1TO8	1	Values for sensors 5-8 (one byte each)	0	(reserved)

The measurement values are given according to the configured sensor group resolution (see *Parameter Set* (page 9)).

Note: This is a legacy command. The maximum value is 255 (1 Byte). Higher bits are not transmitted.

2.6.4 CMD_GET_DATA_9TO16

This command is used to request the readings of sensors 9 to 16.

Command ID: `+0`

CMD_GET_DATA_9TO16	0	0	0	0	0	0	0

Answer (two parts) IDs: `+4`, `+5`

CMD_GET_DATA_9TO16	0	Values for sensors 9-12 (one byte each)	0	(reserved)
CMD_GET_DATA_9TO16	1	Values for sensors 13-16 (one byte each)	0	(reserved)

The measurement values are given according to the configured sensor group resolution (see *Parameter Set* (page 9)).

Note: This is a legacy command. The maximum value is 255 (1 Byte). Higher bits are not transmitted.

2.6.5 CMD_WRITE_PARASET

Use this command to transfer a complete parameter set to the USBoard-USS5. The parameters are stored volatile, which means that they will be lost when the board is switched off. To conveniently configure the board, use the graphical parameter editor.

After transmitting the parameter set, it will immediately be used by the board.

Command ID: `+0`

The command consists of nine messages sent one after the other, each containing some bytes of the parameter set.

CMD_WRITE_PARASET	0	Bytes 1 to 6
CMD_WRITE_PARASET	1	Bytes 7 to 12
CMD_WRITE_PARASET	2	Bytes 13 to 18
CMD_WRITE_PARASET	3	Bytes 19 to 24
CMD_WRITE_PARASET	4	Bytes 25 to 30
CMD_WRITE_PARASET	5	Bytes 31 to 36
CMD_WRITE_PARASET	6	Bytes 37 to 42
CMD_WRITE_PARASET	7	Bytes 43 to 48
CMD_WRITE_PARASET	8	Bytes 49 to 54

Answer ID: +8

Each message is answered individually. The first 8 answers are identical

CMD_WRITE_PARASET	0	0	0	0	0	0	0

and the last answer contains the sum of the first 48 bytes in the parameter set.

CMD_WRITE_PARASET	low byte	high byte	0	0	0	0	0

2.6.6 CMD_WRITE_PARASET_TO_EEPROM

Use this command to write a complete parameter set into the board's EEPROM. The parameters are stored non-volatile, which means that they will be used again the next time the USBoard-USS5 is switched on. To comfortably configure the board, use the graphical parameter editor.

The parameter set will immediately be used after transmission.

Command ID: +0

The command works identically to *CMD_WRITE_PARASET* (page 14), the only difference is the use of the command byte CMD_WRITE_PARASET_TO_EEPROM.

Answer ID: +9

The answer is identical to the answers to *CMD_WRITE_PARASET* (page 14), the only difference is the use of the command byte CMD_WRITE_PARASET_TO_EEPROM.

2.6.7 CMD_READ_PARASET

Use this command to read a complete parameter set from the USBoard-USS5.

Command ID: +0

CMD_READ_PARASET	0	0	0	0	0	0	0

Answer ID: +6

The answer consists of nine messages sent one after the other, each containing some bytes of the parameter set.

CMD_READ_PARASET	0	Bytes 1 to 6
CMD_READ_PARASET	1	Bytes 7 to 12
CMD_READ_PARASET	2	Bytes 13 to 18
CMD_READ_PARASET	3	Bytes 19 to 24
CMD_READ_PARASET	4	Bytes 25 to 30
CMD_READ_PARASET	5	Bytes 31 to 36
CMD_READ_PARASET	6	Bytes 37 to 42
CMD_READ_PARASET	7	Bytes 43 to 48
CMD_READ_PARASET	8	Bytes 49 to 54

2.6.8 CMD_GET_ANALOGIN

Use this command to acquire the data of the four analog inputs.

Command ID: `+0`

CMD_GET_ANALOGIN	0	0	0	0	0	0	0

Answer ID: `+7`

Because the resolution of the on-board AD-converter is 12 bit, the first part of the answer is made up of the four low bytes. Bytes 5 and 6 contain the upper 4 bits of the four channels.

CMD_GET_ANALOGIN	low 8 bit channel 0	low 8 bit channel 1	low 8 bit channel 2	low 8 bit channel 3	high 4 bits of channel 0 and 1	high 4 bits of channel 2 and 3	0

2.6.9 CMD_GET_DATA

This command is used to request the readings of selected sensors. The selection is done by a bit mask where bit 0 stands for group 0, bit 1 stands for group 1 and so on. There will be one answer message per selected group.

Command ID: `+0`

CMD_GET_DATA	(selected groups)	0	0	0	0	0	0

Answer ID: `+(13 + group_id)`

CMD_GET_DATA	(info)	lower 8 bit sensor 0	lower 8 bit sensor 1	lower 8 bit sensor 2	lower 8 bit sensor 3	high 4 bits of sensors 0 and 1	high 4 bits of sensors 2 and 3

The *info* byte contains

- 2 bits encoding the group id (0 to 3)
- 2 bits encoding the sensor resolution (0 for 1 cm, 1 for 0.5 cm, 2 for 0.25 cm, 3 for 0.125 cm)
- 4 bits denoting the sending sensor, either `0xFF` if all sensors are sending or the bit of the active sensor set to 1 for cross echo mode

The *lower 8 bit* byte of a sensor can also contain an error code:

- 0 indicates that the sensor is not physically connected to the USBoard-USS5.

- 1 indicates that an object is closer than the minimum range of the sensor.
- 2 indicates that the sensor is active and sending pulses but has not received any echo. It usually occurs when the object is too far.

2.7 ROS Node

This node handles the communication of the Neobotix USBoard-USS5.

The USBoard-USS5 node has been tested with:

- ROS Kinetic on Ubuntu 16.04
- ROS Melodic on Ubuntu 18.04
- ROS Noetic on Ubuntu 20.04

Find the code of the ROS node at https://github.com/neobotix/neo_usboard_v2.

2.7.1 Installation

1. Clone the `neo_usboard_v2` repository into your catkin workspace source folder:

```
cd your_catkin_workspace/src
git clone https://github.com/neobotix/neo_usboard_v2.git
```

2. Clone `neo_msgs` into your catkin workspace source folder:

```
git clone https://github.com/neobotix/neo_msgs.git
```

3. Fetch the submodules: `vnx-base`, `pilot-base` and `pilot-usboard`:

```
cd neo_usboard_v2
git submodule update --init
```

4. Install `vnx-base` onto your system.
   - ROS Kinetic:

```
sudo dpkg -i vnx-base/x86_64/vnx-base-1.9.3-x86_64-ubuntu-16.04.deb
```

   - ROS Melodic:

```
sudo dpkg -i vnx-base/x86_64/vnx-base-1.9.3-x86_64-ubuntu-18.04.deb
```

   - ROS Noetic:

```
sudo dpkg -i vnx-base/x86_64/vnx-base-1.9.3-x86_64-ubuntu-20.04.deb
```

5. Compile your workspace:

```
cd your_catkin_workspace
catkin_make
```

2.7.2 Launch

In case of using CAN, the bus needs to be configured first:

```
sudo ip link set can0 down
sudo ip link set can0 type can bitrate 1000000
sudo ip link set can0 up
```

To launch the USBoard-USS5 ROS node use:

```
roslaunch neo_usboard_v2 neo_usboard_v2.launch
```

2.7.3 Parameters

The following parameters can be changed according to your needs in `neo_usboard_v2.yaml`:

Parameter	Value	Note
can_id	1024	Needs to be a multiple of 32.
can_device	None or can0	
serial_port	/dev/ttyUSB0	
can_baud_rate	1000000 (bit/s)	Needs to match what is configured on the board.
update_rate	5 (Hz)	Relevant only in transmit mode “Request”, see USBoardV2 GUI.

Note: If `can_device` is set, the CAN bus will be used for communication, otherwise the serial interface specified in `serial_port` is used.

The following parameters can be changed from the parameter server once the ros node is executed.

Parameter	Value	Note
low_pass_gain	1	low pass filter gain (1 = no filtering)
enable_analog_input	false	if to enable reading analog inputs
enable_legacy_format	false	if to use old message format (when transmitting automatically)
enable_can_termination	false	if to connect CAN bus termination resistance on the board
relay_warn_blocked_invert	false	if to invert warn relay output when a sensor is blocked
relay_alarm_blocked_invert	false	if to invert alarm relay output when a sensor is blocked
active_sensors(0 to 15)	true	Active sensors from 1-16
warn_distance(0 to 15)	100 cm	Warning distances of sensors 1-16
alarm_distance(0 to 15)	30 cm	Alarm distances
enable_transmission(0 to 4)	true	if the group transmits in continuous mode
fire_interval_ms	20 ms	time between pulses
sending_sensor	0	index of the sensor which will send the pulse (cross echo mode)
cross_echo_mode	false	if to enable cross echo mode

Note: For setting the ros parameters from the application, it is advised to utilize the */usboard_v2/set_parameters* service. If it is required to set the parameters from the command line, then please utilize the *rosrun dynamic_reconfigure dynparam set param value* command to configure it. Please use the command *ros param list* to see the available parameters.

2.7.4 Topics

The following ROS topics are available:

Name	Type
/usboard_v2/measurements	neo_msgs/msgs/USBoardV2
/usboard_v2/sensor1	sensor_msgs/Range
/usboard_v2/sensor2	sensor_msgs/Range
...	
/usboard_v2/sensor16	sensor_msgs/Range

2.7.5 Multiple USBoards

In case of using multiple USBoard-USS5 simultaneously, it is possible to start multiple ROS nodes, one for each board.

2.7.5.1 Parameters

Each ROS node needs its own yaml configuration file, see for example `neo_usboard_v2.yaml` and `neo_usboard_v2_1.yaml` in https://github.com/neobotix/neo_usboard_v2/tree/main/launch:

```
can_id: 1024
can_device: None
#can_device: can0
serial_port: /dev/ttyUSB0
can_baud_rate: 1000000
update_rate: 5
topic_path: /usboard_v2
```

```
can_id: 1056
can_device: None
#can_device: can1
serial_port: /dev/ttyUSB1
can_baud_rate: 1000000
update_rate: 5
topic_path: /usboard_v2_1
```

Note: CAN IDs need to be a multiple of `0x20`, ie. `32` in decimal.

2.7.5.2 Launching Multiple ROS Nodes

The standard launch file `neo_usboard_v2.launch` contains an example on how to start another ROS node:

```
<?xml version="1.0"?>
<launch>
    <!-- Load usboard_v2 params -->
    <rosparam command="load" ns="usboard_v2" file="$(find neo_usboard_v2)/launch/neo_
↪usboard_v2.yaml"/>
    <!-- start usboard_v2 node -->
    <node pkg="neo_usboard_v2" type="neo_usboard_v2" ns="usboard_v2" name="usboard_v2_
↪node" respawn="false" output="screen"/>
```

(continues on next page)

(continued from previous page)

```
    <!-- Load usboard_v2 params for second board -->
    <rosparam command="load" ns="usboard_v2_1" file="$(find neo_usboard_v2)/launch/
↪neo_usboard_v2_1.yaml"/>
    <!-- start usboard_v2 node for second board -->
    <node pkg="neo_usboard_v2" type="neo_usboard_v2" ns="usboard_v2_1" name="usboard_
↪v2_node_1" respawn="false" output="screen"/>

</launch>
```

Simply uncomment the second block to enable a second ROS node.

To launch all configured nodes:

```
roslaunch neo_usboard_v2 neo_usboard_v2.launch
```

2.7.6 Help

If you receive the error `unable to connect to port /dev/ttyUSB0`, run the following command:

```
sudo usermod -a -G dialout $USER
```

and restart your PC.

2.8 ROS 2 Node

This node handles the communication of the Neobotix USBoard-USS5.

The USBoard-USS5 node has been tested with:

- ROS 2 Foxy on Ubuntu 20.04, Rolling and Humble on Ubuntu 22.04

Find the code of the ROS node at https://github.com/neobotix/neo_usboard_v2-2.

Note: We also have extended our support to Galactic and Humble. Please checkout to the corresponding branch as needed.

2.8.1 Installation

1. Clone the `neo_usboard_v2-2` repository into your ros2 workspace source folder:

   ```
   cd your_ros2_workspace/src
   git clone https://github.com/neobotix/neo_usboard_v2-2.git
   ```

2. Clone `neo_msgs` into your catkin workspace source folder:

   ```
   git clone https://github.com/neobotix/neo_msgs2.git
   ```

3. Fetch the submodules `vnx-base`, `pilot-base` and `pilot-usboard`:

```
cd neo_usboard_v2-2
git submodule update --init
```

4. Install `vnx-base` onto your system.
   - ROS Foxy:

     ```
     sudo dpkg -i vnx-base/x86_64/vnx-base-1.9.3-x86_64-ubuntu-20.04.deb
     ```

   - ROS Humble:

     ```
     sudo dpkg -i vnx-base/x86_64/vnx-base-1.9.6-x86_64-ubuntu-22.04.deb
     ```

5. Compile your workspace:

   ```
   cd your_ros2_workspace
   colcon build --symlink-install
   . install/setup.bash
   ```

2.8.2 Launch

In case of using CAN, the bus needs to be configured first:

```
sudo ip link set can0 down
sudo ip link set can0 type can bitrate 1000000
sudo ip link set can0 up
```

To launch the USBoard-USS5 ROS node use:

```
ros2 launch neo_usboard_v2 neo_usboard_v2.launch.py
```

2.8.3 Parameters

The following parameters can be changed according to your needs in `neo_usboard_v2.yaml`:

Parameter	Value	Note
can_id	1024	Needs to be a multiple of 32.
can_device	None or can0	
serial_port	/dev/ttyUSB0	
can_baud_rate	1000000 (bit/s)	Needs to match what is configured on the board.
update_rate	5 (Hz)	Relevant only in transmit mode “Request”, see USBoardV2 GUI.

Note: If `can_device` is set, the CAN bus will be used for communication, otherwise the serial interface specified in `serial_port` is used.

The following parameters can be changed from the parameter server once the ros2 node is executed.

Parameter	Value	Note
low_pass_gain	1	low pass filter gain (1 = no filtering)
enable_analog_input	false	if to enable reading analog inputs
enable_legacy_format	false	if to use old message format (when transmitting automatically)
enable_can_termination	false	if to connect CAN bus termination resistance on the board
relay_warn_blocked_invert	false	if to invert warn relay output when a sensor is blocked
relay_alarm_blocked_invert	false	if to invert alarm relay output when a sensor is blocked
active_sensors(0 to 15)	true	Active sensors from 1-16
warn_distance(0 to 15)	100 cm	Warning distances of sensors 1-16
alarm_distance(0 to 15)	30 cm	Alarm distances
enable_transmission(0 to 4)	true	if the group transmits in continuous mode
fire_interval_ms	20 ms	time between pulses
sending_sensor	0	index of the sensor which will send the pulse (cross echo mode)
cross_echo_mode	false	if to enable cross echo mode

2.8.4 Topics

The following ROS topics are available:

Name	Type
/usboard_v2/measurements	neo_msgs/msg/USBoardV2
/usboard_v2/sensor1	sensor_msgs/msg/Range
/usboard_v2/sensor2	sensor_msgs/msg/Range
...	
/usboard_v2/sensor16	sensor_msgs/msg/Range

2.8.5 Help

If you receive the error `unable to connect to port /dev/ttyUSB0`, run the following command:

```
sudo usermod -a -G dialout $USER
```

and restart your PC.

2.9 Graphical User Interface

Download the GUI here: Windows[4], Linux[5]

2.9.1 Introduction

The USBoard-USS5 comes with a graphical user interface (GUI). This program can be used to configure the USBoard-USS5 for the application it will be used in. It also displays the current sensor readings and helps in setting up the system.

[4] https://neobotix-docs.de/files/USBoardUSS5-GUI_Windows.zip
[5] https://neobotix-docs.de/files/USBoardUSS5-GUI_Linux.zip

The USBoard-USS5-GUI requires a PC with a Microsoft Windows® or Ubuntu LTS operating system and a free USB port. For industrial applications the USB connection can be replaced with a RS-232 interface. In this case the computer will need an RS-232 interface (COM port) as well or a USB-Serial-Converter.

In Windows, the GUI comes with an installer. Just execute it with a double click and follow the instructions. The process should create a shortcut on your desktop that you can double click to start the GUI.

The Linux version comes with Debian packages for the latest Ubuntu LTS versions. Install the appropriate package with:

```
sudo dpkg -i the_right_package.deb
```

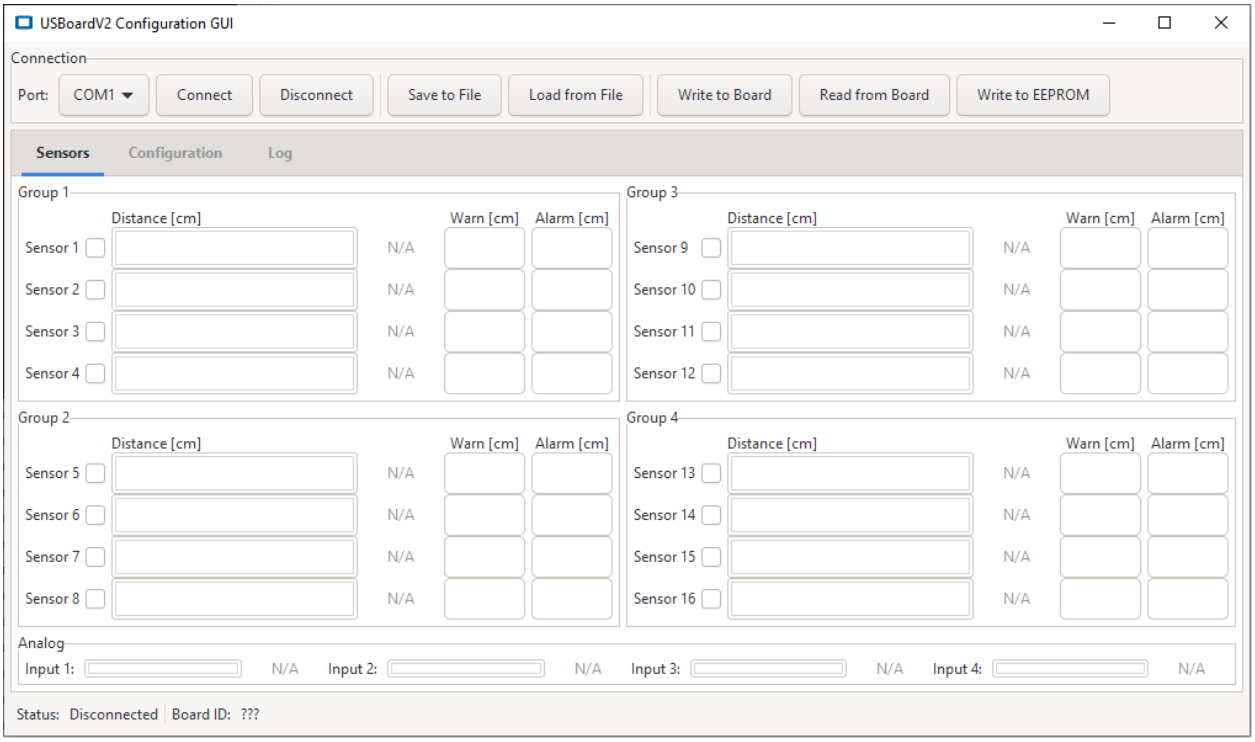
where you replace `the_right_package` with the one that fits your operating system. To run the GUI after the installation, please use the following command:

```
/opt/neobotix/usboard2-gui/bin/usboard2_gui
```

2.9.2 First Steps

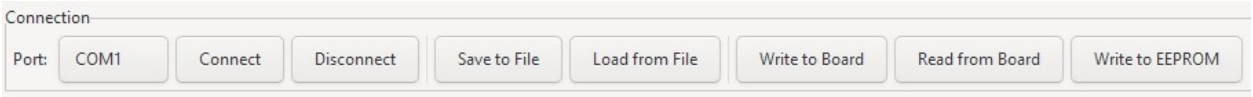
2.9.2.1 The GUI Layout

The graphical user interface is divided into three main areas: the main functions area at the top, the central data area and the footer.



The Functions Area

The functions area is divided into three groups.



On the left you can find the basic functions to establish a connection to the USBoard-USS5 and also to disconnect again.

Port: Here you can select the interface that will be used to communicate with the USBoard-USS5. When plugging in the USB cable the operating system will automatically add a virtual COM port that can be selected here.

Connect Opens the selected interface and establishes a connection to the USBoard-USS5.

Disconnect Closes an open connection to a board.

Use the buttons in the central group to save the current USBoard-USS5 configuration into a file on your local computer or to load a configuration from such a file.

Save to File Saves the configuration from the GUI into a file on your computer.

Load from File Loads a configuration file and changes the GUI settings accordingly.

On the right you can find the buttons to transfer the current settings from the GUI to the USBoard-USS5 and to read the board's configuration into the GUI.

Write to Board Transfers the parameters to the volatile memory of the USBoard-USS5. After the next restart, the old parameters from the non-volatile memory will be used again.

Read from Board Reads the current configuration from the USBoard-USS5 and displays it in the GUI.

Write to EEPROM Writes the parameters into the non-volatile memory of the USBoard-USS5. These settings will be activated after a restart of the board.

The Data Area

The central data area features three different views that can be activated by clicking on the three tabs on top.

Sensors This view visualises the readings of the active sensors and displays their warning and alarm thresholds.

Configuration Here you can adjust the communication parameters and configure the sensors' behaviour.

Log The log lists system messages related to certain events.

The Footer

The footer displays the current connection state and the serial number of the board that the GUI is currently connected to.

2.9.2.2 Visualizing Measurements

Ultrasonic Sensors

The ultrasonic sensors are grouped by four, with each group being displayed in its own box within the data area.

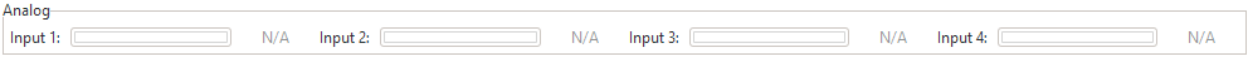
The check boxes next to the sensor numbers indicate whether a sensor is active. Only active sensors are triggered and read by the USBoard-USS5.

Tip: Deactivating sensor channels that are not in use will reduce the cycle time for reading all sensors and update measurements faster. Deactivated channels will be skipped by the USBoard-USS5 without waiting for an echo.

The warning and alarm thresholds of each sensor are displayed on the far right. The current distance measurements are shown as coloured bars and numerical values in the centre of the box.

Note: The USS5 sensors have an upper and a lower *measurement limit* (page 6). If an object is detected closer than the lower limit the message "BLOCKED" is shown instead of the measurement value. If no object can be detected within the measurement range at all the message "TOO FAR" will be shown.

Analogue Inputs



The measurements of the analogue inputs can be found below the ultrasonic sensor display. The analogue input values are also shown both as numerical value and as bars. The bars indicate the ratio of the current input voltage relative to the *maximum input voltage* (page 6).

2.9.3 Configuration

2.9.3.1 Basic Features

Sensor Configuration

The basic settings of the ultrasonic sensors can be changed right in the main measurement view.

Activation

Set the tick in the checkbox to activate the according sensor. Deactivated sensors will not be triggered by the USBoard-USS5, will not send an ultrasonic pulse and the board will not wait for an echo.

Tip: Deactivate all channels that are not in use. This will reduce the cycle time for handling all sensors and provide updated measurements as fast as possible.

If a sensor is not physically connected to the USBoard-USS5 but is marked as active the GUI will indicate this case by displaying “N/A” instead of a measurement value.

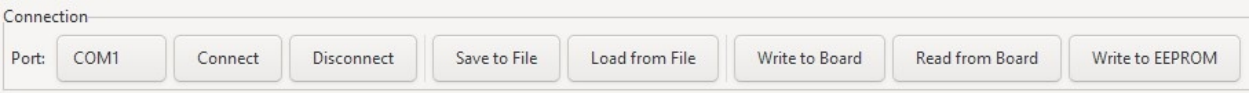
Thresholds

Every sensor has one warning and one alarm threshold. As soon as any active sensor measures a distance that is below the threshold the corresponding relay will be switched. The relays remain in active state as long as the measurement of at least one sensor is below its associated limit. Setting a threshold value of 0 will deactive the function for this sensor.

Only integer values are accepted, indicating the distance in centimetres.

Handling Configurations

You can use the buttons in the functions area to save configurations to your local computer or to reuse them later. When connected you can also transfer a new configuration to a USBoard-USS5 or read the current one from the board.

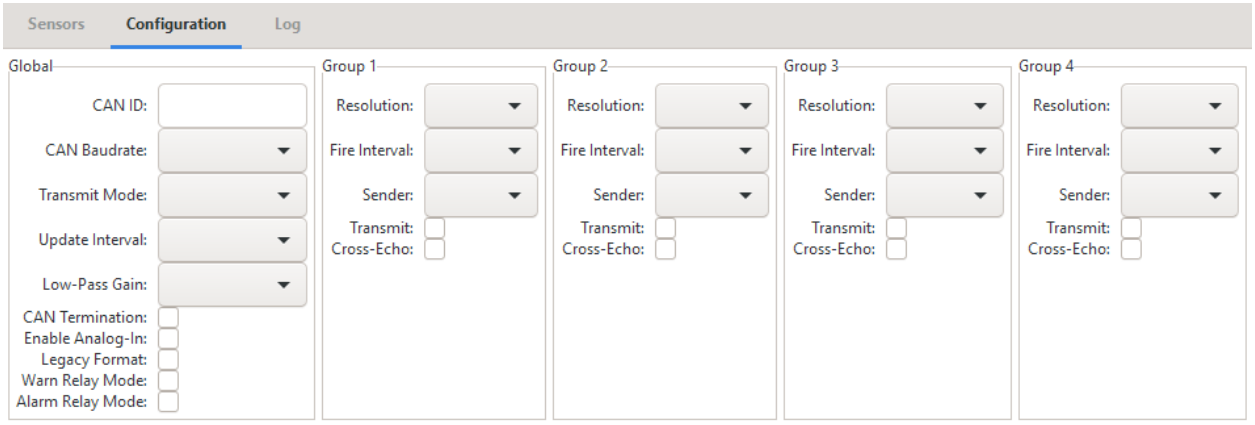


Most of these functions are self-explanatory but some details should be kept in mind.

- Whenever you transfer a configuration (from file to GUI, from GUI to USBoard-USS5 or the other way round) the data on the receiving end will be overwritten. There is no Undo function.
- Use Write to Board to quickly test new settings. After restarting the USBoard-USS5 these changes will be lost and the configuration from the EEPROM memory will be used instead.
- Using Write to EEPROM stores the configuration in the non-volatile memory and also applies it immediately.
- Read from Board will only read the settings that are currently in use. These may differ from the configuration stored in the EEPROM.

2.9.3.2 Advanced Features

Please select the tab `Configuration` to change the view in the data area.



Global Settings

The box `Global` on the left contains communication and general settings.

CAN ID The base address of the USBoard-USS5 for CAN bus communication.

CAN Baudrate The CAN bus data rate.

Transmit Mode Sets the output mode of the board. Request means that the USBoard-USS5 will only send messages on request. In all other settings it will send out messages automatically via the selected interface.

Update Interval Defines the cycle time for sending messages in automatic sending mode.

Low-Pass Gain The low pass filter feature can be used to smoothen the measurements and to prevent single, potentially faulty, readings from immediately switching the relays. Each new measurement of a sensor is added to the output value according to the filter's weight. A setting of 0.3 means that the output is calculated from 70% of the previous value and 30% of the new measurement. A setting of 1.0 means that each new measurement is directly used as output value while the old value is discarded, in effect deactivating the filter.

CAN Termination In case of the USBoard-USS5-IP the CAN bus terminating resistor can be activated here. This parameter is without function in the normal USBoard-USS5.

Enable Analog-In Activates the use of the four analogue inputs. If your application requires very short cycle times and does not use these inputs it is recommended to disable the analogue inputs completely.

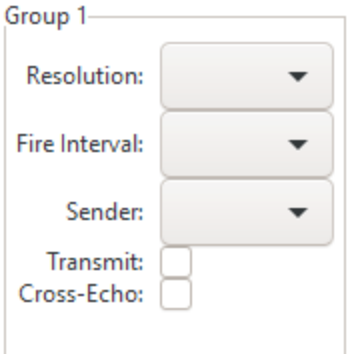
Legacy Format Enables the use of the USBoard V1 protocol. This makes the USBoard-USS5 compatible to older applications but disables some features.

Warn Relay Mode When checked, blocked sensors do not affect the warning relay, which can still be switched by the remaining functional sensors. Use this mode only if blocked sensors do not reduce the operational safety of your system. If unchecked, the warning relay is switched on as soon as a single sensor is blocked, regardless of the readings measured by the other sensors. Use this mode to indicate all cases of blocked sensors.

Alarm Relay Mode Equivalent to Warn Relay Mode but affecting the alarm relay.

Sensor Group Settings

The ultrasonic sensors are grouped in sets of four sensors each. The settings in this area always apply to all sensors of a group. Different groups can have completely different configurations.



Resolution The measurement data can be sent in four different resolutions: 1 cm, 0.5 cm, 0.25 cm or 0.125 cm.

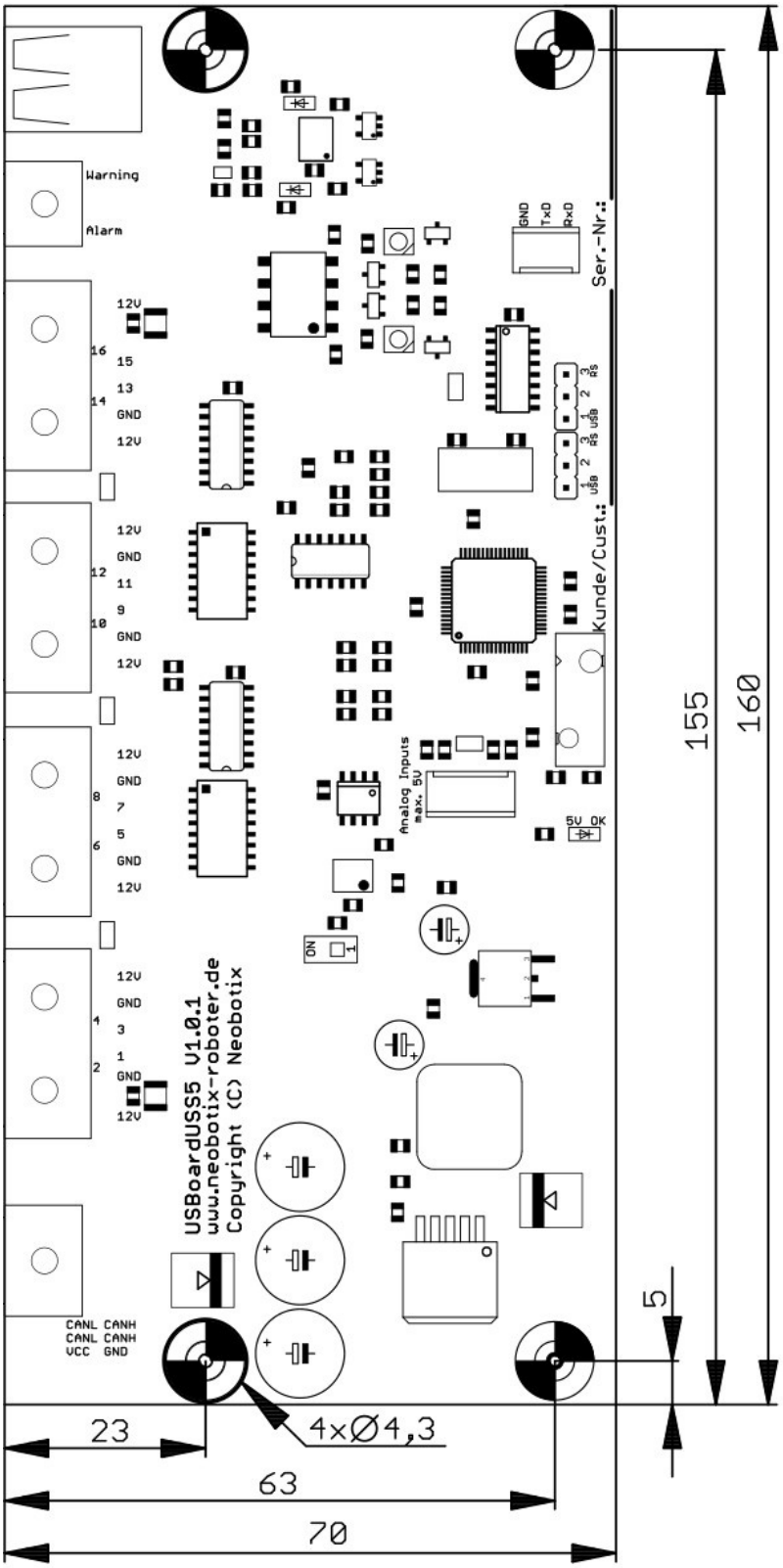
Fire Interval The time between firing two sensors can be adjusted to avoid unwanted echos from earlier pulses. The minimum time a sensor needs to send a pulse and evaluate the echo is 10 ms. The minimum firing interval of 10 ms offers the fastest cycle time and is designed for a relatively short distance detection up to 150 cm. When the object is placed at a distance of more than 150 cm, there is not enough time for the sensor to receive the reflected echo within 10 ms. To provide enough time for the reflected echo to be received by the sensor at the maximum distance the default firing interval is 20 ms.

Sender If the group operates in cross-echo mode the pulse sending sensor can be selected here.

Transmit Includes the group into the data output.

Cross-Echo Switches the group from normal, sequential operation mode to cross-echo mode.

2.10 Dimensions



Note: Move `S1` to position `ON` to activate the CAN terminating resistor.

Attention: The mounting holes near the supply connector X1 are electrically connected to the ground plane of the USBoard-USS5. Please insulate this mounting point if necessary in your installation.

The USBoard-USS5 can be mounted in any orientation.

2.11 Connector Descriptions

Note: More information about these connectors can be found at *Connectors* (page 38).

2.11.1 X1

Würth Elektronik MPC3, 2 rows, 6 pins

Pin	Description
1	Power Supply Input
2	CAN High
3	CAN High
4	Ground
5	CAN Low
6	CAN Low

Pins 2 and 3 are bridged, as well as pins 5 and 6. This allows two CAN cables (CAN-in and CAN-out) to be connected easily.

2.11.2 X2 - X5

Würth Elektronik MPC3, 2 rows, 12 pins

Pin	Description
1	Sensor supply voltage
2	Ground
3	Sensor data line (X2: Sensor 2, X3: Sensor 6, X4: Sensor 10, X5: Sensor 14)
4	Sensor data line (X2: Sensor 4, X3: Sensor 8, X4: Sensor 12, X5: Sensor 16)
5	Ground
6	Sensor supply voltage
7	Sensor supply voltage
8	Ground
9	Sensor data line (X2: Sensor 1, X3: Sensor 5, X4: Sensor 9, X5: Sensor 13)
10	Sensor data line (X2: Sensor 3, X3: Sensor 7, X4: Sensor 11, X5: Sensor 15)
11	Ground
12	Sensor supply voltage

2.11.3 X6

Würth Elektronik MPC3, 2 rows, 4 pins

Pin	Description
1	Alarm relay output (normally open)
2	Warning relay output (normally open)
3	Alarm relay output (normally open)
4	Warning relay output (normally open)

2.11.4 X7

TE Connectivity HE14, 1 row, 4 pins

Pin	Description
1	Analogue input 1
2	Analogue input 2
3	Analogue input 3
4	Analogue input 4

2.11.5 USS5

Molex MX64, 3 pins, key A

Pin	Description
1	Supply
2	Signal
3	Ground

2.12 Additional Parts

2.12.1 Connectors

An overview of the connectors can be found at *Connectors* (page 38).

2.12.2 USBoard-USS5 Configuration Cable

The basic cable set (order number X211) speeds up the configuration of the USBoard-USS5 and first tests. It consists of:

- Connector X1: Connections for power supply (1 m, flying leads, red = plus, black = minus) and CAN (1 m, female D-Sub 9 connector, no terminating resistor, pin 2 - CAN Low, pin 7 - CAN High)
- USB cable: 1,5 m, USB 2.0, A-type to B-type

This cable set can also be manufactured by the customer using the connectors described at *Connector Descriptions* (page 30).

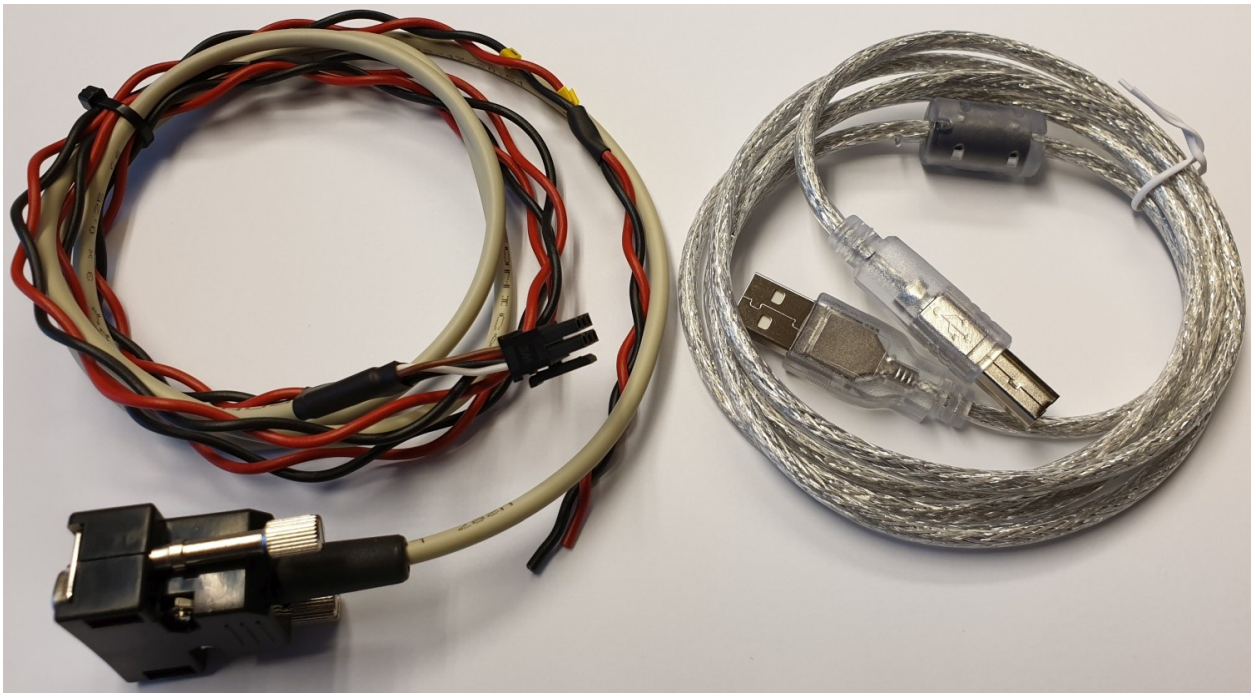


Fig. 1: USBoard-USS5 configuration cable (X211)

2.12.3 Sensor Cable Sets

The USBoard-USS5 features four connectors for cable sets that connect four sensors each.

Neobotix offers unshielded cable sets in different lengths for first tests and prototyping. The default version (order number X214) is 2 m long and can be plugged into any of the USBoard-USS5's four sensor connectors X2 to X5. Other cable lengths (up to 25 m) and designs are available on request.

It is recommended to manufacture cable sets that meet your individual requirements when moving to the actual application.

2.12.4 Connector Sets

Neobotix offers sets of crimp contacts and connector housing for customers who plan to manufacture the cable sets by themselves and only need the correct parts. These contacts can be crimped with generic tools well enough for first tests and prototyping.

The connector set for the USBoard-USS5 (order numer X212) contains mating crimp contacts and housing for all connectors of the USBoard-USS5 except the USB cable.

The set for the ultrasonic sensors (order number X213) contains the crimp contacts and housing for one sensor.

2.12.5 Ultrasonic Sensors

Ultrasonic sensors Bosch USS5.0 with radial connector are available under order number X210.

Fig. 2: Cable set for 4 x USS5 sensors (X214)

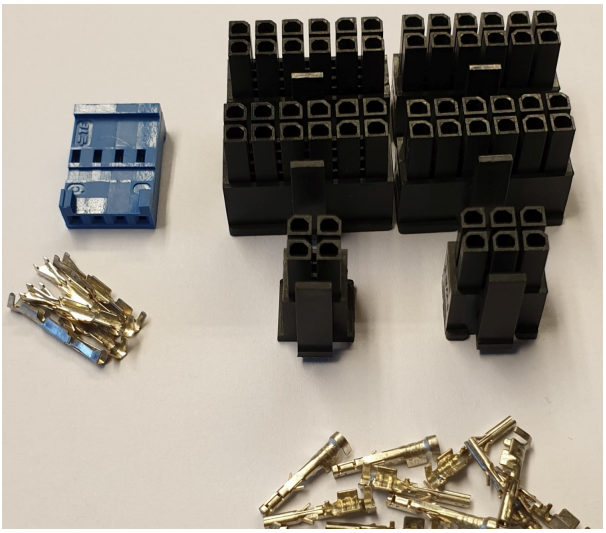


Fig. 3: Connector set for USBoard-USS5 (X212)

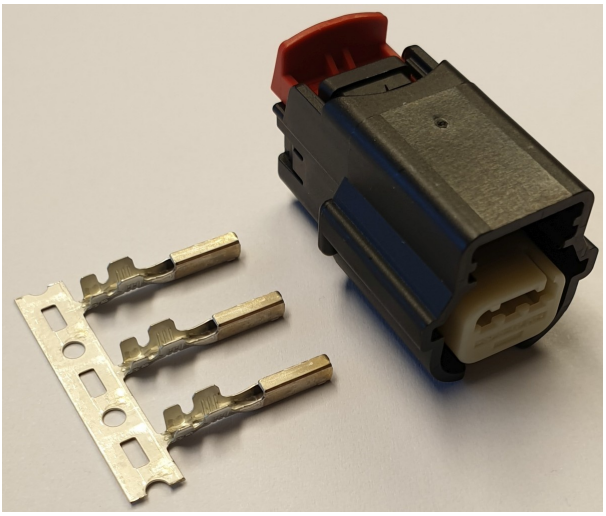
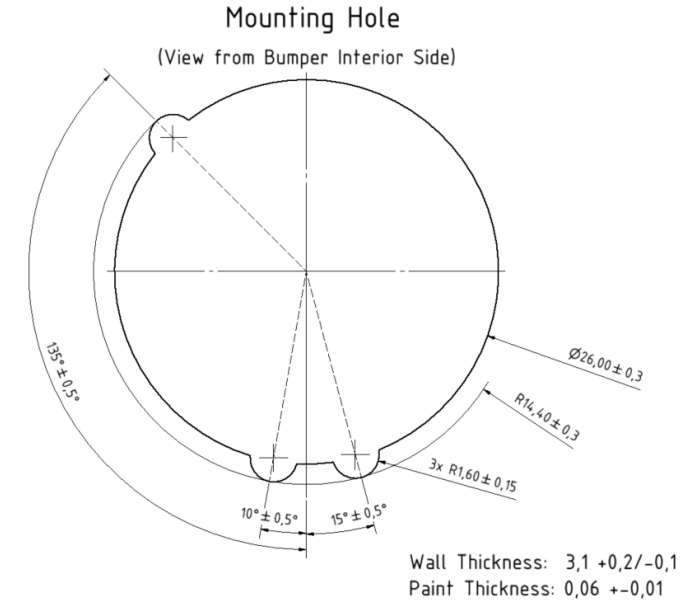


Fig. 4: Connector set for USS5 (X213)

Fig. 5: Ultrasonic sensor Bosch USS5.0 (X210)

2.12.6 USS5 Mounting Frames

The USS5 mounting kit (order number X216) is used to mount the sensor on a front panel.



2.13 FAQ

Note: This section is being extended as we get feedback from customers. If your specific questions is not yet listed please either take a look into the full documentation on these pages or get in touch with us directly. We will do our best to help you!

2.13.1 General Questions

2.13.1.1 How do the USBoard-USS5 and ultrasonic sensors work?

Both the basic principle of ultrasonic sensors and the cross-echo operating mode of the USBoard-USS5 are described in detail at *Operating Principle* (page 4).

2.13.1.2 The sensor readings are not what I expected.

Unfortunately this is a common problem of ultrasonic sensors and caused by the physical measuring principle. Different from laser light ultrasound waves are easily disturbed and bounced around which leads to faulty readings in some situations. They also react differently to different materials and surfaces. This is the reason why the USBoard-USS5 is not a safety device.

Please take a look into our *FOV tests* (page 7). We also strongly recommend to do some tests and experiments with the system before going into production.

2.13.1.3 How to get measurement data as fast as possible?

Disabling all sensor channels that are currently not in use will prevent the USBoard-USS5 from trying to send pulses on these channels and from waiting for an echo. This will reduce the cycle time needed for updating measurements from all sensors that are actually used. Please take a look at the parameter set description at *Parameter Set* (page 9) and command `CMD_SET_CHANNEL_ACTIVE` at *Commands* (page 13).

In some applications it might also be an option to use the *Cross Echo Mode* (page 5).

2.13.2 Connection Problems

2.13.2.1 The CAN bus reports errors

Please make sure that there is at least one CAN terminating resistor on the bus. The recommended configuration is one 120 Ohm resistor between CAN High and CAN Low at each physical end of the bus line.

The USBoard-USS5 features *a micro switch* (page 29) to enable the integrated terminating resistor. Please use the GUI to activate the resistor of the USBoard-USS5-IP.

2.13.2.2 The GUI does not connect to the USBoard-USS5

In case of a direct USB connection please check the following:

- Did you select the correct port? Some computers still have internal COM ports that can be mixed up with the virtual port of the USBoard-USS5's USB connection. Try with all available ports.
- Has the port number changed? Different USBoard-USS5 may have different port numbers assigned by the operating system. Please close and restart the GUI to make it update the port list.

2.14 Legal Notes

The general legal notes can be found at *Legal Notes* (page 2).

2.14.1 EU Declaration of Conformity

This product fulfils all relevant directives of the European Union.

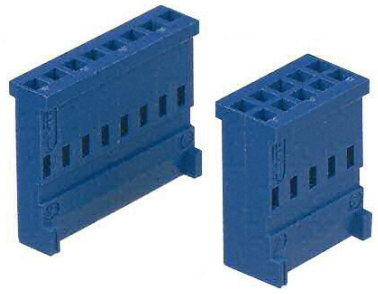
2.14.2 RoHS Information

This product complies to the RoHS directives 2011/65/EU (RoHS 2) and 2015/863/EU of the European Parliament and the Council on the restriction of the use of hazardous substances in electrical and electronic equipment.

CHAPTER 3

Connectors

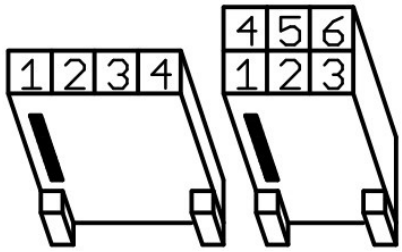
3.1 TE Connectivity - HE14



Pins	TE Connectivity	Farnell	RS Components
3 pins, 1 row	281838-3	429582	532-333
4 pins, 1 row	281838-4	429594	532-349
5 pins, 1 row	281838-5	429600	532-355
6 pins, 2 rows	281839-3	429650	532-406
8 pins, 2 rows	281839-4	429661	532-412
10 pins, 2 rows	281839-5	429673	532-428
12 pins, 2 rows	281839-6	429685	532-434

Crimp contacts	TE Connectivity	Farnell	RS Components
AWG 28-24	182734-2	429715	532-456

In Neobotix products the pin assignment of the HE14 connectors is as shown below.



3.2 Würth Elektronik - MPC4

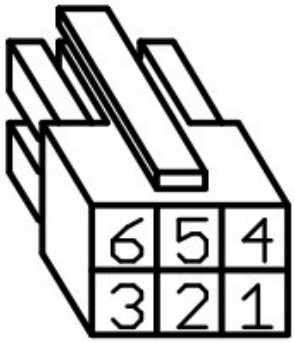
Please check the Würth Elektronik online catalogue[6] for details on the MPC4[7].

Pins (in 2 rows)	Würth Elektronik
2	649002113322
4	649004113322
6	649006113322
8	649008113322
10	649010113322
16	649016113322

Crimp contacts	Würth Elektronik
AWG 24-18	64900613722

In Neobotix products the pin assignment of the MPC4 connectors is as shown below.

[6] https://www.we-online.com/en/products/components/overview
[7] https://www.we-online.de/katalog/de/em/connectors/wire-to-board/wr_mpc4/



3.3 Würth Elektronik - MPC3

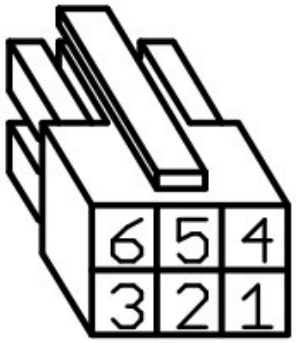
Please check the Würth Elektronik online catalogue[8] for details on the MPC3[9].

Pins (in 2 rows)	Würth Elektronik
4	662004113322
6	662006113322
12	662012113322

Crimp contacts	Würth Elektronik
AWG 24-20	66200113722

In Neobotix products the pin assignment of the MPC3 connectors is as shown below.

[8] https://www.we-online.com/en/products/components/overview
[9] https://www.we-online.de/katalog/de/em/connectors/wire-to-board/wr_mpc3/


6 5 4
3 2 1

CHAPTER 4

Qualified Personnel

This product must only be modified, commissioned, operated and serviced by qualified personnel. Qualified personnel are defined as persons who

- due to their specialist training and experience have adequate knowledge for the work at hand,
- have been instructed by the responsible robot operator in the operation of the robot or its parts and the currently valid safety guidelines,
- are sufficiently familiar with the applicable official health and safety regulations, directives and generally recognised engineering practice (e.g. DIN standards, VDE stipulations, engineering regulations from other EC member states) that they can assess the work safety aspects of the product and
- have access to this manual and who have read it.

The following groups of persons are generally not considered qualified:

- Employees, interns or other academic staff not familiar with the product,
- visitors and quests,
- all members of other departments of the company or institution in which the product is operated.

This list is not intended to be exhaustive.